Database Security Issues

This module is the final of four modules that describe the use of the Trusted Database Interpretation (TDI) of the Trusted Computer System Evaluation Criteria (TCSEC) for database product evaluation and certification. The basic terms and concepts presented in the TDI are summarized in TDI Module One. Module Two describes database security policies that can be supported by a trusted database management system (DBMS). TDI Module Three describes various architectural approaches for building a trusted DBMS. This module describes other database security issues that are not covered by the TDI but are important issues in database security, such as inference, aggregation, and database integrity.

Module Learning Objectives

This module provides the student with information on security issues which are unique to database management systems (DBMSs) and the underlying security policy issues which impact DBMSs. Upon completion of this module, the student should:

1) be familiar with inference, and aggregation of data and how these issues affect database design and classification.

2) be familiar with the relationships between integrity and secrecy in a DBMS, especially between entity integrity, referential integrity, and signaling channels.

Overview

This module discusses some security issues of special importance in databases: inference, aggregation, and database integrity. The overview provides a brief introduction to these subjects. More details on many of these topics are covered are given in a set of TDI Companion Documents [DINF92], [DAGG92], [DINT92]. The module is divided into various sections. The overview section contains a discussion of the underlying security policy and classification management issues which create some of the aggregation and inference problems in DBMSs. The next three sections overview specific issues.

There are three fundamental concepts associated with security issues in DBMSs: 1) how to classify output at the proper level based on the associated security policy; 2) how to approach the labeling issues associated with a specific data element; and 3) how to determine the appropriate downgrading of the information in the database.

Current DoD policy requires that information be correctly classified based on content. As stated in the introduction by G.W. Smith in Research Directions in Database Security “What should not happen is that a report, which only contains unclassified data, be classified as TOP SECRET just because it ‘touched’ a TOP SECRET environment. To accomplish this, data (and the

associations between data) must be classified at a level that accurately reflect its real classification."

To understand the associated security issues, the student must also understand policy guidance for data classification. DoD 5200.1-R provides the basic guidance for classification of sensitive information. Based on this guidance, developers and system integrators determine the specific database design to comply with these policies. Inherent in the implementation of databases based on DoD classification policies may be instances of inference, and data aggregation. The following sections highlight these specific issues. It must be emphasized that these issues are still the subject of research.

Inference

Inference occurs when a user is able to make a valid deduction, based only on data the user is authorized to access, about data which the user is not authorized to access. The TDI companion document *"Inference Problems in Multilevel Secure Database Management Systems"* [DINF92] provides an in-depth description of the problem and some discussion on potential solutions. Inference can be grouped into three types:

1. Inference based on data derived from low data which is retrieved from the database by a user.

2. Inference based on data derived from low data in conjunction with metadata (integrity or value constraints) stored in the database.

3. Inference which requires some external knowledge in addition to that found in the database.

Inference can occur regardless of the specific granularity associated with the security levels in a database. Whether security is assigned to relations, tuples, columns, or the actual data element, inference can still occur.

The first type of inference is generally a database design issue. A user simply requests data from the system and is granted access to data which allows the user to infer more sensitive information. For example, suppose a database contains a relation giving the rank of employees and a tuple giving the salary associated with a rank. If a user can observe employee-rank pairs and rank-salary pairs, the user can infer the salary for an employee. If the salary for an employee is to be kept from the user, the user must be prevented from retrieving either employee-rank or rank-salary tuples.

The second circumstance where a user can infer information about higher level data in a database occurs when database integrity constraints involve data at multiple levels. In such a case, if the low level user attempts to update information at a lower level but is prevented from doing so because of a constraint involving higher level data, the user can discover something about the higher level data. An example is a database containing cargo data for airplanes. Suppose the database has a relation with airplane number, cargo

item, and cargo weight. If there is a constraint that the total cargo weight for an airplane must be less than some number, a user can infer the weight of more sensitive cargo by attempting to insert cargo items at the level for which the user is authorized. If the insert is denied, the user can determine that the weight of the more sensitive cargo items is greater than the total allowed weight minus the total weight of the items the user can see.

The third type occurs when users can infer information for which they are not authorized based on the information for which they are authorized access and facts they already know. For example, if a user is only authorized access to unclassified flights but knows that fifty planes have been scheduled for the week and only forty-five are visible to his terminal, he may infer from his knowledge that five flights are classified.

Aggregation

Aggregation occurs when a collection of individual pieces of information must be classified at a higher level than any single piece of information which comprises it. Aggregation is sometimes grouped into *cardinal aggregation* and *data association.* In cardinality aggregation the *number* of data objects is the issue. For example, suppose the phone number of any one individual in an organization is considered unclassified, but a sufficiently large set of phone numbers is considered classified. Some researchers have suggested (e.g., Seaviews [JLUN89]) that in cardinal aggregation problems, the data objects which make up the aggregate set should be handled as if classified at the level of the aggregate, but a special mechanism should allow facilitated downgrade of individual data objects in the aggregate. This view is based on an analysis of the example above in which the phone book itself is classified but individuals with legitimate access to the phone book are authorized to release name-phone number pairs from the book.

In data association, the problem is that the association between data objects is considered sensitive, but the individual objects are not sensitive. Data association is sometimes viewed as an instance of inference since the user may be able to infer the association by executing queries returning the individual data objects. At any rate, this problem can typically be solved through proper database design by appropriately classifying tuples which allow the sensitive association to be established.

Systems enforcing standard MAC and DAC policies can be implemented using generalized mechanisms that are application independent. No such mechanism is likely to work for general inference and aggregation control. Inference and aggregation tend to be unique and specific to the individual databases so no generic solution is practical. Inference and aggregation control are problems inherent in the data being managed, not problems inherent in the data management software of the DBMS (although certain DBMS mechanisms can be used to reduce this problem). The potential for inference and aggregation make the proper design and object classification of MLS databases a difficult problem even when access to the database is controlled by a trusted DBMS. The TDI companion document "*Aggregation Problems in Multilevel*

Secure Database Management Systems" [DAGG92] provides an in-depth description of the aggregation problem and some discussion on potential solutions.

Database Integrity

A major problem with MLS databases is that data integrity and data secrecy are frequently at odds with one another. Enforcing integrity to ensure that the relationships between data objects are not invalidated as a result of data insertion, modification, or deletion poses specific security issues in MLS systems. In particular, enforcing integrity constraints can cause inference problems and may provide signaling channels (i.e., covert channels). In general, a database integrity constraint requires that a relationship be maintained over certain sets of data elements in the database. If the set covered by an integrity constraint includes data at multiple levels, knowledge of the results of attempts to update those elements in the set which are accessible to a user together with knowledge of the constraint itself can yield knowledge of the elements not directly accessible to the user. This is an example of inference. Since a high level process can alter sensitive data elements in the set, this inference can then be used as a channel to signal information about other sensitive data.

Entity integrity and referential integrity are two important integrity constraints that are enforced by a relational DBMS. The TDI Companion Document, *Entity and Referential Integrity in Multilevel Secure Database Management Systems*, [DINT92], discusses these topics in detail. The following sections provide an overview of the relationships of entity and referential integrity to secrecy.

Entity Integrity

Entity integrity requires that the primary key be unique (actually an inherent property of a "key") and all attributes of the primary key be non-null. Data in a relation represents some entity in the real world which must have a unique identification. The primary key performs this function. The primary key must also be non-null because it would be a contradiction in terms to use some entity that did not have any unique identification.

Enforcing entity integrity provides inference. Specifically, if a user attempts to insert a tuple with a given key and the insert is rejected because a more sensitive tuple already exists with the same primary key, the user can infer the existence of a sensitive tuple with the given primary key. This can be used as a signaling channel by a high level process which inserts and deletes tuples with particular primary keys signaling a low user who attempts to insert low tuples with the same primary keys. Since the "alphabet" of the channel is the set of possible values for the primary key, this can be quite a large bandwidth channel.

A solution to this problem is polyinstantiation. Polyinstantiation refers to multiple instantiations of a tuple with the same primary key. When a user

attempts to insert a tuple with the same primary key as an existing more sensitive tuple, the insert is allowed, and the new tuple is inserted at the user's current access class. Essentially this can be viewed as considering the access class of the key (or in the case of tuple granularity classification the access class of the tuple) as part of the primary key. Since there will only be one instance of a tuple with a given primary key at a given access class, entity integrity is preserved with this concept of an expanded primary key. In this approach entity integrity is changed from a multilevel constraint to a single level constraint. In practice there are frequently problems with using polyinstantiation. There needs to be an understanding of which instance represent the "right" tuple for a polyinstantiated tuple. User may get misleading results from queries. Whether and how to use polyinstantiation depends on the specific database and how it is used.

Since many applications using a relational database assume that the requirements above hold for primary keys (i.e., all attributes in the primary key are non-null), in most cases it is necessary that these requirements are met when the database is viewed from any specific access class. This has two consequences. First, the primary key must be uniformly classified. Otherwise, when viewed from the access class of an element in the primary key not dominated by the access classes of some other attribute of the primary key, some of the primary key attributes would appear to be null. Second, the access class of all attributes not in the primary key must dominate the access class of the primary key. Otherwise, when viewed from the access class of an element not dominated by the access classes of all attributes of the primary key, some of the primary key attributes would appear to be null.

Referential Integrity

Referential integrity based on the ANSI standard for SQL2 requires that each referencing foreign key value must have an identical target primary key value in the referenced relation. Extending the concept of referential integrity from single-level relations to multilevel relations adds significant complexity. This is because restrictions are needed to enforce referential integrity controls in MLS DBMSs without compromising secrecy. These problems arise when access classes of the foreign key and the referenced primary key are not the same.

When tuples are inserted, deleted, or updated, the DBMS must ensure that there is still a primary key of the value referenced by each foreign key. There are several ways in which this can be done. The requested action can be denied if it would violate referential integrity or additional actions can be automatically performed to guarantee that referential integrity is preserved. For example, suppose a user tries to delete a tuple which is referenced by another tuple through a foreign key. If the delete were performed without any additional action, the foreign key would then reference a non-existent tuple resulting in a violation of referential integrity. Possible solutions are to have the DBMS prohibit the deletion of the tuple as long as it is referenced by other tuples, to delete the tuple as requested and also delete all tuples referencing the deleted tuple, or to delete the tuple as requested and change the references of other tuples to some default value. Each of these approaches introduces the

possibility of signaling channels when the access classes of the referencing key and the referenced key differ.

In order to maintain referential integrity when the database is viewed from any specific access class there are two requirements on the access class of the foreign key and the referenced primary key. First, for the reasons described in the discussion of entity integrity, both the foreign key and the referenced primary key should be uniformly classified (i.e., all attributes included in the key should have the same access class).

The second requirement follows from the standard referential integrity requirement that there must exist a primary key of the value referenced by the foreign key. Therefore, when the database is viewed from the access class of the foreign key, the referenced primary key must be visible. Since the only elements visible from the access class of the foreign key are at access classes dominated by the access class of the foreign key, this yields the requirement: The access class of the foreign key must dominate the access class of the primary key of the referenced tuple.

Beyond these requirements on the classification of the foreign key and referenced primary key, the need to enforce Mandatory Access Control imposes restrictions on the methods the DBMS can use to enforce referential integrity when a user requests an update, insert, or delete. Consider, for example, the case discussed above where a user tries to delete a tuple which is referenced by another tuple through a foreign key. If the foreign key is at an higher access class than the access class of the referenced tuple this would allow the user to infer the existence of a foreign key which the user could not directly access. This represents a signaling channel. Deleting or altering the tuple with the foreign key would not result in information compromise but might create problems with the usefulness of the data depending upon the particular database. These issues are discussed in considerable detail in [DINT92].

In today's environment, it is not possible to obtain both perfect data integrity and perfect multilevel security. Furthermore, the industry trend is towards supporting greater integrity enforcement capabilities. A variety of methods are described in [JMAI91], which show the extremes of maintaining either complete integrity or the prospect of complete security, as well as some intermediate tradeoff point which may allow greater latitude to particular applications in choosing an appropriate balance. Integrity is one of the principle benefits of a DBMS, but the use of the integrity features must be carefully scrutinized for non-obvious security flaws. There are several research projects directed at solutions for this problem. Examples of issues which are the focus of various research efforts are: 1) the covert channel problem (which becomes significant when the MLS system requires the mechanism to write-up to enforce the integrity constraint), and 2) use of auditing of significant events.

Required Readings

TLUN92 Research Directions in Database Security, Teresa Lunt, Editor, Springer Verlag, 1992. [Chapters 11, 12, & 13]

Chapter 11 provides the concept of classification and why MLS databases are an issue.

Chapter 12 is an overview on semantics of data classification which is the basis for the Bell-LaPadula model.

Chapter 13 provides the views on data inference and aggregation issues.

DINF92 National Computer Security Center, *Inference Problems in Multilevel Secure Database Management Systems*, Part of the TDI Companion Documents Series, May 1992.

This report provides a state-of-the-art survey of the various aspects of the inference problem. It defines and characterizes the inference problem as it relates to multilevel secure database systems and describes methods that have been developed for resolving the problem.

DAGG92 National Computer Security Center, *Aggregation Problems in Multilevel Secure Database Management Systems*, Part of the TDI Companion Documents Series, May 1992.

This document is intended to stimulate discussion and research: it is not intended to serve as a guideline. The following subjects are covered: data aggregation problem, relationship between inference and data aggregation, and current measures for data aggregation control in commercial and system-high databases. An overview of on-going research projects are included.

DINT92 National Computer Security Center, *Entity and Referential Integrity in Multilevel Secure Database Management Systems*, Part of the TDI Companion Documents Series, May 1992.

This document defines integrity constraints in the context of single-level systems and then extends them to MLS systems. Various referential integrity rules have been investigated with both secrecy and integrity in mind.

Other Related Readings

JMAI91 B. Maimone, R. Allen, "*Methods for Resolving the Security vs. Integrity Conflict*", 4th RADC Database Security Workshop, April 1991.

TLUN92 Research Directions in Database Security, Teresa Lunt, Editor, Springer Verlag, 1992. [Chapters 2, 8, and 20]

JHIN88 T. H. Hinke, "*Inference Aggregation Detection in Database Management Systems*", 1988 IEE Symposium on Security and Privacy, April 1988.

JLUN88 T. F. Lunt, et. al., *"Final Report Vol 1: Security Policy and Policy Interpretation for a Class A1 Multilevel Secure Relational Database*

System," Computer Science Laboratory, SRI International, Menlo Park, California, 1988.

JLUN89 T. F. Lunt, "*Aggregation and Inference: Facts and Fallacies*", 1989 IEEE Symposium on Research in Security and Privacy, May 1989.